Urządzenia Peryferyjne Laboratorium

22.11.2023

Grupa nr 7B, Czwartek, godz. 11:00, tydzień nieparzysty

Autorzy:

264488 Adam Czekalski

252560 Karol Szydlak

nr	Treść zadania	data wykonania
3.	Ćwiczenie 8 Karta dźwiękowa. Zapisywanie i odtwarzanie dźwięku.	9/11/2023

1. Wstęp teoretyczny

1.1 Zasady zapisywania dźwięku

Dźwięk jest falą mechaniczną rozchodzącą się w ośrodku sprężystym, jego reprezentacja cyfrowa polega na próbkowaniu i kodowaniu tej fali. Podczas próbkowania amplituda fali dźwiękowej jest mierzona w regularnych odstępach czasu, standardowa częstotliwość próbkowania dla audio CD to 44,1 kHz (44,100 próbek na sekundę).

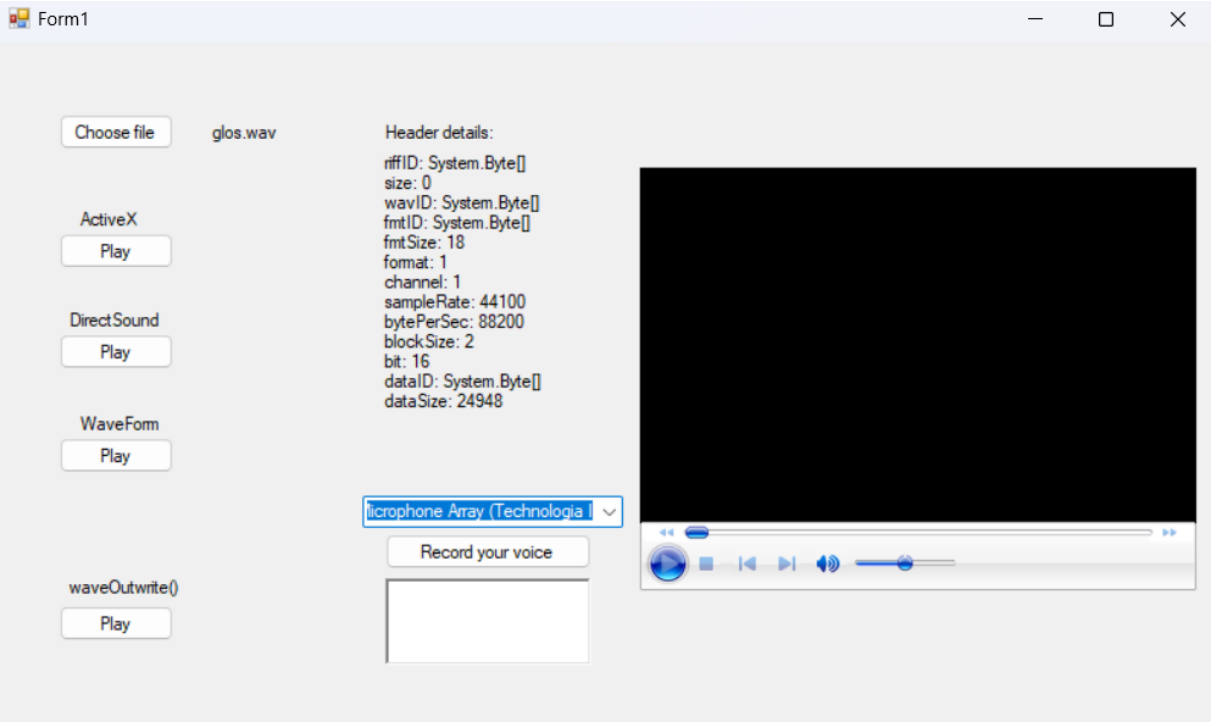
- Technologia wykorzystywane do zapisu i odtwarzania dźwięku
  -Mikrofon – przetwornik elektroakustyczny przetwarzający falę dźwiękową na przemienny prąd elektryczny.
  -Karta dźwiękowa – urządzenie przetwarzające sygnał dźwiękowy z formy analogowej na cyfrową lub odwrotnie, zawierają interfejsy audio umożliwiające podłączanie do nich - urządzeń, takich jak mikrofony czy głośniki
  -Głośnik – przetwornik elektroakustyczny przekształcający prąd elektryczny w falę dźwiękową
- Parametry zapisu dźwięku
  -Częstotliwość próbkowania – określa liczbę próbek rejestrowanych na sekundę podczas nagrywania, wyższa wartość oznacza lepszą jakość i większy rozmiar pliku
  -Głębokość bitowa – ilość bitów używanych do zapisu jednej próbki
  -Liczba kanałów – określa liczbę niezależnie nagrywanych ścieżek dźwiękowych (mono-jedna ścieżka, stereo-dwie).

1.2 Przykładowe formaty zapisu informacji dźwiękowej do zbioru

- WAV – popularny format plików dźwiękowych stosowany w Windows, przechowuje dźwięk w postaci nieskompresowanej
- MP3 – format kompresji stratnej, pozwala na zmniejszenie rozmiaru pliku przy pogorszeniu jakości dźwięku
- M_4A – format zgodny ze standardem MP4, może zawierać dźwięk skompresowany stratnie lub bezstratnie.

2. Aplikacja

Aplikacja do obsługiwania karty dźwiękowej została napisana w języku C#, korzystając z biblioteki SlimDX (DirectSound), Windows Multimedia (winmm.dll) i NAudio (WaveForm) oraz z Windows Forms w celu stworzenia interfejsu graficznego.



Rysunek 2-1 Interfejs aplikacji

2.1. Odtworzenie dźwięku za pomocą ActiveX

```
1 reference
private void axPlay_Click(object sender, EventArgs e)
{
    axWindowsMediaPlayer1.URL = filepath; //przypisz ścieżkę do pliku i odtwórz dźwięk
}
```

Rysunek 2-2 Metoda pozwalająca na odtwarzanie dźwięku za pomocą ActiveX po kliknięciu guzika

Aby odtworzyć dźwięk tym sposobem, skorzystano z komponentu „COM” Windows Media Player. Umieszczono odtwarzacz w aplikacji okienkowej i przypisano mu ścieżkę do pliku.

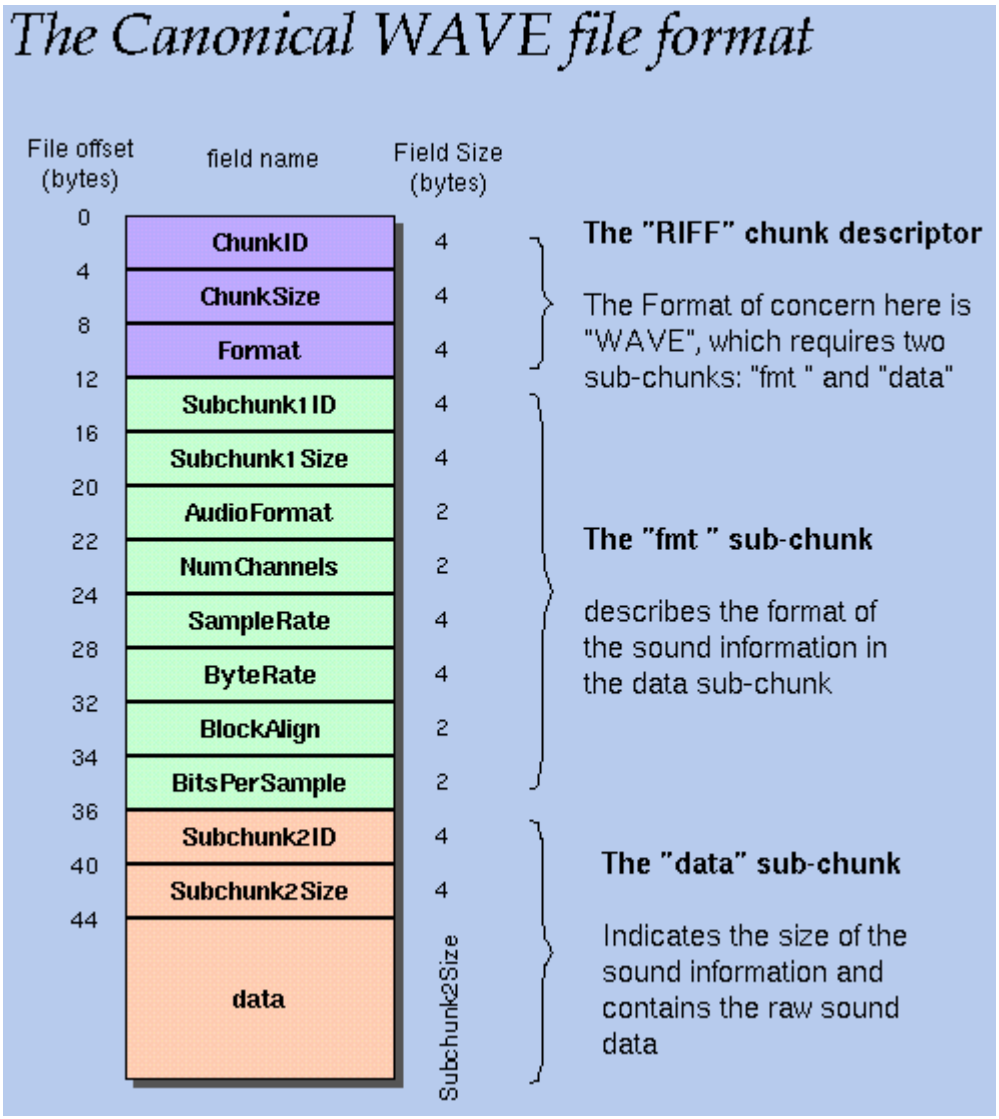
2.2. Odtworzenie dźwięku za pomocą komendy PlaySound()

```
[DllImport("winmm.dll")] //import metody z biblioteki Windows Multimedia
1 reference
private static extern bool PlaySound(string sound, IntPtr hmod, uint flags);

1 reference
public void PlayWithPlaySound(string filepath)
{
    bool success = false;

    Task.Run(() =>
    {
        success = PlaySound(filepath, IntPtr.Zero, 0); //odtworzenie dźwięku
    }
    );

    if (!success)
    {

        Console.WriteLine("Error playing sound");
    }
}
```

Rysunek 2-3 Fragment klasy WaveFormPlaying.cs, w której znajduje się metoda PlaySound()

Metodę PlaySound() zaimportowano z biblioteki Windows Multimedia. Przekazywane do niej są: ścieżka do pliku dźwiękowego, parametr hmod ustawiony na IntPtr.Zero w celu przeszukania zasobów systemu związanych z dźwiękiem oraz parametr flags ustawiony na 0, żeby odtworzyć dźwięk bez żadnych modyfikacji. Dodatkowo, odtwarzanie dźwięku jest przeprowadzane w osobnym wątku, aby nie zawiesić okna aplikacji.

2.3. Odczytanie i wyświetlenie poszczególnych pól nagłówka WAV



Rysunek 2-4 Struktura nagłówka pliku z rozszerzeniem .wav

```
1 reference
public string GetInfo(string filepath)
{
    var fileStream = new FileStream(filepath, FileMode.Open, FileAccess.Read);
    var binaryReader = new BinaryReader(fileStream);

    try
    {
        riffID = binaryReader.ReadBytes(4);
        riffSize = binaryReader.ReadUInt32();
        wavID = binaryReader.ReadBytes(4);
        fmtID = binaryReader.ReadBytes(4);
        fmtSize = binaryReader.ReadUInt32();
        audioFormat = binaryReader.ReadUInt16();
        channels = binaryReader.ReadUInt16();
        samplesPerSec = binaryReader.ReadUInt32();
        bytesPerSec = binaryReader.ReadUInt32();
        blockAlignment = binaryReader.ReadUInt16();
        bitsPerSample = binaryReader.ReadUInt16();
        dataID = binaryReader.ReadBytes(4);
        dataSize = binaryReader.ReadUInt32();

    }
    finally
    {
        binaryReader.Close();
        fileStream.Close();
    }

    return ToString();
}
```

Rysunek 2-5 Metoda sczytująca informacje z nagłówka .wav, znajdująca się w klasie WaveHeader.cs

Nagłówek pliku .wav jest sczytywany poprzez czytanie kolejnych bajtów nagłówka. W nagłówku można znaleźć następujące informacje:

- riffID – określa typ bloku w pliku
- riffSize – określa rozmiar reszty bloku w bajtach
- wavID – określa format pliku
- fmtID – określa identyfikator podbloku zawierający informacje o formacie dźwięku
- fmtSize – określa rozmiar podbloku
- audioFormat – określa format dźwięku
- channels – określa czy jest to dźwięk mono czy stereo
- samplesPerSec – częstotliwość próbkowania dźwięku
- bytesPerSec – określa ilość bitów na próbkę
- blockAlignment – określa ilość bajtów na jedną próbkę wszystkich kanałów
- bitsPerSample – określa ilość bitów przypisanych do każdego kanału
- dataID – określa identyfikator podbloku, który zawiera dane dźwiękowe
- dataSize – określa rozmiar podbloku z danymi dźwiękowymi

2.4. Odtworzenie dźwięku za pomocą DirectSound

Aby to zrealizować, skorzystano z biblioteki SlimDX, która obsługuje DirectSound.

```
WaveFormat format = new WaveFormat();
//pobranie informacji z nagłówka
format.BitsPerSample = (short)waveHeader.bitsPerSample;
format.BlockAlignment = (short)waveHeader.blockAlignment;
format.Channels = (short)waveHeader.channels;
format.FormatTag = (WaveFormatTag)waveHeader.audioFormat;
format.SamplesPerSecond = (int)waveHeader.samplesPerSec;
format.AverageBytesPerSecond = format.SamplesPerSecond * format.BlockAlignment;
```

Rysunek 2-6 Przekazanie danych z nagłówka .wav

Na początku tworzony jest obiekt klasy WaveFormat, a następnie zostają przypisane do niego poszczególne własności nagłówka pliku .wav .

```
SoundBufferDescription desc = new SoundBufferDescription(); //obiekt opisujący pierwszy bufor dźwiękowy
desc.Format = format; //przypisanie wcześniej utworzonego obiektu WaveFormat
desc.Flags = BufferFlags.GlobalFocus; //ustawienie bufora dźwięku żeby był aktywny wszędzie
desc.SizeInBytes = 8 * format.AverageBytesPerSecond; //obliczenie rozmiaru bufora w bajtach

PrimarySoundBuffer pBuffer = new PrimarySoundBuffer(ds, desc); //główny bufor dźwiękowy

SoundBufferDescription desc2 = new SoundBufferDescription(); //obiekt opisujący drugi bufor dźwiękowy
desc2.Format = format;
//ControlPositionNotify - żeby bufor obsługiwał powiadomienia o zmianie pozycji odtwarzania dźwięku (np. jak się zakończy)
//GetCurrentPosition2 - żeby można było z niego pobrać informację o aktualnym położeniu w pliku
desc2.Flags = BufferFlags.GlobalFocus | BufferFlags.ControlPositionNotify | BufferFlags.GetCurrentPosition2;
desc2.SizeInBytes = 8 * format.AverageBytesPerSecond;

SecondarySoundBuffer sBuffer1 = new SecondarySoundBuffer(ds, desc2); //dodatkowy bufor
```

Rysunek 2-7 Utworzenie dwóch buforów dźwiękowych i ustawienie ich parametrów

Następnie tworzone są dwa bufory dźwiękowe – w tym jeden główny.

```
NotificationPosition[] notifications = new NotificationPosition[2]; //punkty powiadomień
notifications[0].Offset = desc2.SizeInBytes / 2 + 1; //pierwszy punkt na środku długości bufora dźwiękowego
notifications[1].Offset = desc2.SizeInBytes - 1; ; //drugi punkt na końcu długości bufora dźwiękowego

notifications[0].Event = new AutoResetEvent(false); //żeby się sam nie restartował
notifications[1].Event = new AutoResetEvent(false);
sBuffer1.SetNotificationPositions(notifications); //ustaw pozycje tych dwóch punktów
```

Rysunek 2-8 Punkty powiadomień

Kolejnym krokiem jest utworzenie „punktów” powiadamiających o aktualnym położeniu w pliku dźwiękowym, aby wiedzieć czy odtworzono już połowę dźwięku czy też dźwięk został odtworzony w całości.

```
byte[] bytes1 = new byte[desc2.SizeInBytes / 2]; //do odczytu/zapisu pierwszej połowy danych z pliku do bufora
byte[] bytes2 = new byte[desc2.SizeInBytes]; //do jednorazowego odczytu i zapisu całego bufora na początku przetwarzania

Stream stream = File.Open(audioFile, FileMode.Open); //otwarcie strumienia do odczytu pliku dźwiękowego

Thread fillBuffer = new Thread(() => {
    int bytesRead;

    bytesRead = stream.Read(bytes2, 0, desc2.SizeInBytes); //czytaj bajty z pliku dźwiękowego
    sBuffer1.Write<byte>(bytes2, 0, LockFlags.None); //zapisz te bajty w dodatkowym buforze dźwiękowym
    sBuffer1.Play(0, PlayFlags.Looping); //i odtwórz dźwięk

    while (true)
    {
        if (bytesRead == 0) { break; }
        notifications[0].Event.WaitOne(); //oczekiwanie na osiągnięcie połowy długości pliku dźwiękowego
        bytesRead = stream.Read(bytes1, 0, bytes1.Length);
        sBuffer1.Write<byte>(bytes1, 0, LockFlags.None);

        if (bytesRead == 0) { break; }
        notifications[1].Event.WaitOne(); //oczekiwanie na osiągnięcie końca pliku dźwiękowego
        bytesRead = stream.Read(bytes1, 0, bytes1.Length);
        sBuffer1.Write<byte>(bytes1, desc2.SizeInBytes / 2, LockFlags.None);
    }
    stream.Close(); //zamknięcie strumienia pliku
    stream.Dispose();
});
fillBuffer.Start();
```

Rysunek 2-9 Odtwarzanie dźwięku w wątku

Następnie są tworzone 2 tablice na dane z pliku i rozpoczynane jest przetwarzanie danych z pliku w wątku.

2.5. Zarejestrowanie dźwięku za pomocą mikrofonu i zapisanie dźwięku w zbiorze WAV

Rejestracja dźwięku z mikrofonu została zrealizowana za pomocą biblioteki NAudio.

```
1 reference
private void recordButton_Click(object sender, EventArgs e)
{
    //zmienna isRecorded pozwala naprzemiennie rozpoczynać i zatrzymywać nagrywanie
    //w zależności od tego czy wcześniej je rozpoczęto czy zatrzymano
    if (!isRecorded)
    {

        wave = new WaveIn();
        wave.WaveFormat = new NAudio.Wave.WaveFormat(44100, 1); //ustawienie częstotliwości próbkowania dźwięku na 44100Hz oraz trybu mono
        wave.DeviceNumber = comboBox1.SelectedIndex; //wybór mikrofonu do przechwytywania z listy
        //obsługa zdarzeń
        wave.DataAvailable += Wave_DataAvailable; //zdarzenie dostarczające nowe dane dźwiękowe do przetworzenia
        wave.RecordingStopped += Wave_RecordingStopped; //zakończenia nagrywania

        writer = new WaveFileWriter(outputFileName, wave.WaveFormat); //ustalenie ścieżki na zapisanie dźwięku
        wave.StartRecording(); //rozpoczęcie nagrywania

        isRecorded = !isRecorded;
    }

    else
    {
        wave.StopRecording(); //zatrzymanie nagrywania
        isRecorded = !isRecorded;
    }
}
```

Rysunek 2-10 Nagrywanie dźwięku

Po wybraniu urządzenia wejściowego i naciśnięciu przycisku „Record your voice” zostaje rozpoczęta rejestracja głosu z mikrofonu. Jeśli przycisk zostanie wciśnięty po raz drugi, dźwięk zostanie zapisany w lokalizacji określonej przez ścieżkę w zmiennej *outputFileName*.

3. Źródła

- https://www.researchgate.net/figure/The-structure-of-wav-file-format_fig1_273630623
- https://pl.wikipedia.org/wiki/G%C5%82o%C5%9Bnik
- https://pl.wikipedia.org/wiki/Mikrofon
- https://pl.wikipedia.org/wiki/D%C5%BAwi%C4%99k
- https://pl.wikipedia.org/wiki/WAV
- https://pl.wikipedia.org/wiki/MP3
- https://pl.wikipedia.org/wiki/M4A